LISTING OF THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Original) A system comprising:

a first node having an associated cache including data having an associated first cache state, the first cache state being capable of identifying the first node as being an ordering point for serializing requests from other nodes for the data.

2. (Original) The system of claim 1, wherein the first cache state enables the first node to provide a data response to a request for the data from a second node for the data without updating a system memory.

3. (Original) The system of claim 1, wherein the first cache state enables the first node to provide an ownership data response to a request for the data from a second node, the ownership data response transferring the ordering point from the first node to the second node.

4. (Original) The system of claim 3, wherein the first node provides the ownership data response without updating a system memory.

5. (Original) The system of claim 3, wherein the first node defines a first processor and the second node defines a second processor, each of the first processor and the second processor having an associated cache, the associated caches of the first and second processors each comprising a plurality of cache lines, each cache line having a respective tag address that identifies associated data and each cache line having state information that indicates a state of the associated data for the respective cache line, the first and second processors being capable of communicating with each other and with other nodes of the system through an interconnect.

6. (Original) The system of claim 5, further comprising a first cache controller associated with the first processor and a second cache controller associated with the second processor, the first cache controller being operative to manage data requests and responses for

the associated cache of the first processor, the first cache controller effecting state transitions associated with the data in the associated cache of the first processor based on the data requests and responses for the associated cache of the first processor, the second cache controller being operative to manage data requests and responses for the associated cache of the second processor, the second cache controller effecting state transitions associated with the data in the associated cache of the second processor based on the data requests and responses for the associated cache of the second processor.

7. (Original) The system of claim 5, wherein the system implements a hybrid cache coherency protocol wherein each of the first and second processors employs a source broadcast-based protocol to issue a request for the data and employs an associated forward progress protocol to reissue a request for the data if the request fails in the source broadcast protocol.

8. (Original) The system of claim 7, wherein the forward progress protocol comprises a null-directory protocol.

9. (Original) The system of claim 7, wherein the source broadcast protocol comprises an incomplete protocol.

10. (Currently Amended) The system of claim 1, wherein the first node comprises a cache including a plurality of cache lines, the system being capable of assigning a cache state to each of the cache lines to identify the status of data cached in the cache line, the cache state being selected from the group consisting of:

a cache state indicating that the data is not cached in the cache line;

a cache state indicating that the data cached in the cache line is valid and unmodified, that other nodes may have valid cached copies of the data, and that the node associated with the cache line cannot respond to snoops by returning a copy of the data;

a cache state indicating that the data cached in the cache line is valid and unmodified, that the data cached in that cache line is the only cached copy of the data in the system, and that the node associated with the cache line can respond to snoops by returning a copy of the data;

a cache state indicating that the data cached in the cache line is valid and unmodified, that other nodes may have valid copies of the data, and that the node associated with the cache line can respond to snoops by returning a copy of the data;

a cache state indicating that the data cached in the cache line is valid and more up-to-date than a copy of the data stored in a system memory, that the data cached in the cache line has not been modified by the node associated with the cache line, that the data cached in the cache line is the only cached copy of the data in the system, that the node associated with the cache line can respond to snoops by returning a copy of the data, and that the node associated with the cache line writes the data back to memory upon displacement;

a cache state indicating that the data cached in the cache line is valid and has been modified, that the data cached in the cache line is the only cached copy of the data in the system, that the node associated with the cache line can respond to snoops by returning the data, and that the node associated with the cache line writes the data back to memory upon displacement;

a cache state indicating that the data cached in the cache line is valid and more up-to-date than the copy of the data stored in system memory, that the node associated with the cache line cannot modify the data cached in the cache line, that other nodes may have valid copies of the data in the cache line, that the node associated with the cache line can respond to snoops by returning the data, and that the node associated with the cache line writes the data back to memory upon displacement; and

a cache state indicating that the cache line is transitioning between cache states.

11. (Currently Amended) A multi-processor network comprising:

a plurality of processor nodes, each processor node having at least one associated cache;

the plurality of processor nodes employing a coherency protocol, the coherency protocol employing ordering points for serializing requests for data associated with the at least one associated cache of the plurality of processor nodes, the ordering point for the data is ~~capable of being~~ associated with the at least one associated cache of one of the processor nodes.

12. (Original) The multi-processor network of claim 11, wherein the coherency protocol employs a first cache state at a first node of the plurality of processor nodes to identify the

first node as an ordering point for the data, the first cache state enabling the first node to provide a data response to a request from a second node for the data without updating a system memory.

13. (Currently Amended) The multi-processor network of claim 11, wherein ~~the~~ an ownership data response transfers the ordering point from the first node to the second node.

14. (Currently Amended) The multi-processor network of claim 11, the ~~system~~ multi-processor network being capable of assigning a cache state to each associated cache of each processor node to identify the status of a block of data in the associated cache, the cache state being selected from the group consisting of:

a cache state indicating that the block of data does not exist in the associated cache;

a cache state indicating that the block of data in the associated cache is valid and unmodified, that other processor nodes may have valid copies of the block of data, and that the processor node associated with the associated cache cannot respond to snoops by returning a copy of the block of data;

a cache state indicating that the block of data in the associated cache is valid and unmodified, that the block of data in the associated cache is the only cached copy of the block of data in the multi-processor network, and that the processor node associated with the associated cache can respond to snoops by returning a copy of the block of data;

a cache state indicating that the block of data in the associated cache is valid and unmodified, that other nodes may have valid copies of the block of data, and that the processor node associated with the associated cache can respond to snoops by returning a copy of the block of data;

a cache state indicating that the block of data in the associated cache is valid and more up-to-date than a copy of the block of data stored in a system memory, that the block of data in the associated cache has not been modified, that the block of data in the associated cache is the only cached copy of the block of data in the system, and that the processor node associated with the associated cache can respond to snoops by returning a copy of the block of data and writes the block of data back to memory upon displacement;

a cache state indicating that the block of data in the associated cache is valid and has been modified, that the block of data in the associated cache is the only cached copy of the block of data in the system, and that the processor node associated with the associated cache

can respond to snoops by returning a copy of the block of data and writes the block of data back to memory upon displacement;

a cache state indicating that the block of data in the associated cache is valid and more up-to-date than the copy of the block of data stored in system memory, that the processor node associated with the associated cache cannot modify the block of data, that other processor nodes may have valid copies of the block of data, and that the processor node associated with the associated cache can respond to snoops by returning a copy of the block of data and writes the block of data back to memory upon displacement; and

a cache state indicating that the associated cache is transitioning between cache states.

15. (Original) A system comprising:

means for employing a cached ordering point to serialize requests for a block of data from nodes of the system; and

means for associating the cached ordering point for the block of data with a first node of the system.

16. (Original) The system of claim 15, wherein the means for associating the ordering point comprises means for assigning a first cache state to the first node of the system.

17. (Original) The system of claim 15, further comprising means for providing a data response from the first node to a request from a second node for the data without updating a system memory.

18. (Original) The system of claim 15, further comprising means for providing an ownership data response from the first node to a request from a second node for the data, the ownership data response transferring the ordering point from the first node to the second node.

19. (Original) A method comprising:

employing a first cache state to identify a first node of a system as being an ordering point for a block of data; and

providing a data response from the first node to a request from a second node of the system for the block of data.

20. (Original) The method of claim 19, further comprising enabling the ordering point to provide the data response without updating a system memory.

21. (Original) The method of claim 19, wherein providing a data response comprises:

providing an ownership data response; and

transferring the ordering point from the first node to the second node.

22. (Original) The method of claim 19, wherein providing a data response comprises:

employing a source broadcast protocol to deterministically resolve the request from the second node for the block of data; and

employing a forward progress technique to deterministically resolve the request from the second node for the block of data if the request from the second node cannot be deterministically resolved through employing the source broadcast protocol.

23. (Original) The method of claim 22, wherein employing a forward progress technique comprises employing a forward progress protocol.

24. (Original) A coherency protocol operative to assign a cache state to a cache line of one node of a plurality of nodes of a system, the cache state defining the one node as an ordering point in the system for data in the cache line of the one node.